Implementation of Hierarchical Algorithms for Line extraction, Texture discrimination and Optical flow estimation on a Pyramidal DSP Architecture

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Shaikh Intekhab-e-Aalum

to the

DEPARTMENT OF ELECTRICAL ENGINEERING,
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

March 1997

EE-1997-M-AAL-IMP

CERTIFICATE

This is to certify that the work contained in this M. Tech thesis entitled **Implementation of Hierarchical Algorithms for Line extraction, Texture discrimination, and Optical flow estimation on a Pyramidal DSP Architecture** *has been carried out by* **Shaikh Intekhab-e-Aalum** *under my supervision and has not been submitted elsewhere for a degree.*

[illegible]
Administrative Supervisor
Dr. Govind Sharma
Associate Professor
Dept. of Electrical Engg.
Indian Institute of Technology
Kanpur.

[illegible]
Supervisor
Dr. Sumana Gupta
Associate Professor
Dept. of Electrical Engg.
Indian Institute of Technology
Kanpur.

ACKNOWLEDGEMENT

With a deep sense of gratitude, I wish to acknowledge my gratefulness to my thesis supervisor, **Dr. Sumana Gupta** for her guidance, invaluable suggestions and encouragement throughout this thesis work.

Shaikh Intekhab-e-Aalum

Contents

List of Figures

Abstract

Research in the area of high speed real-time image/video processing has emphasized the need for parallel processing i.e. parallel architecture and algorithms suitable for parallel implementation. In this thesis we discuss the configuration of a TMS320C40 DSP based real time image processing system where the processors are connected to form a Pyramidal architecture. This thesis also includes the hierarchical algorithms, a class of algorithms suitable for implementation on a pyramidal architecture. The hierarchical algorithms for straight line extraction, texture discrimination and optical flow estimation and their implementation on the hierarchical architecture are discussed. In the line extraction algorithm Hough transform is used for representing the edge pixels. Signal-to-noise ratio of Hough transform is large for smaller image size. Thus in the hierarchical approach the image is divided into blocks and the results from the smaller blocks are combined and transfered to the higher levels in the pyramid. The processor at the apex(highest level) thus contains the description of the largest and strongest edges in the entire image. The texture discrimination algorithm uses the T.E.M.(Texture Energy Map) as the criteria to discern the different textured portions. The T.E.M. provides an initial decomposition of textured images into its main components. The optical flow estimation is based on spatio-temporal gradient approach. It uses the interlevel motion smoothness constraint rather than the local smoothness constraint. The use of image pyramid enables us to estimate large flow vectors. The multigrid technique is employed to speed up the convergence of the algorithm by solving the optical flow problem in different spatial frequency bands(pyramid levels). The results of the implemented algorithms are validated using different sets of images.

Chapter 1

Introduction

Real-time image processing, in general, requires processing of large volume of data at a very high rate . This emphasizes the need for parallel processing architectures as well as algorithms which can be mapped onto such architectures. It is well known that an image often contains large amount of redundant information both in spatial and spectral domain. For real-time image processing, the vision system must avoid or discard excessive details at the earliest stages of analysis so that *s*mart sensing, which requires selective and task oriented gathering of information from the visual world, can be carried out. The pyramidal approach is one of the ways of implementing the above task. In this thesis we consider a hierarchical(Pyramidal) architecture for *R*eal Time Image Processing (RTIPS) based on TMS320C40 DSP processors. The system is very cost-effective and is suitable for implementation of hierarchical algorithms. An image pyramid is a sequence of copies of original image in which resolution and sample density are decreased in regular steps. The lower resolutions of the image pyramid contains less data so that by selecting only the relevent details the processing can be done. The human visual system uses the same strategy [1]. The goal of multiresolution representation is to describe the structure of the image by systematically removing the finer details. Thus the top layer or the apex is related to more *g*lobal abstraction levels such as interpretation of the scene whereas the lower levels contain the *d*etailed information of the image data. We consider the implementation of hierarchical algorithms for Line Extraction, Texture Discrimination and Optical Flow Estimation on the configured RTIPS.

In images containing man made objects, the grey-level discontinuities lie mostly along straight lines. Therefore the detection of straight lines is an important task in many pattern recognition systems. In this thesis the pyramidal algorithm used is based on splitting the complete image into a number of sub-images. At the bottom level of the pyramid short line segments are detected by applying Hough transform to small size sub-images. The algorithm proceeds, bottom up, from this low level description by grouping line segments within local neighborhoods into longer lines. Line segments which have *l*ocal support propagate up the hierarchy and take part in grouping at higher levels. The length of the line determines approximately the level in the pyramid to which it propagates [2]. The highest level thus contains the information of the longest straight lines in the complete image.

Among the low-level vision problems, texture discrimination is certainly one of the most difficult. Despite the fact that textures are quickly preattentively discriminated by a human observer, there is still no appropriate model for textures. The perception of textures depends upon local and not pointwise properties However there is no predefined neighborhood size over which textures can be analyzed. In this thesis the pyramidal algorithm for texture discrimination is discussed. The pyramidal approach is of importance since we have all reduced resolution versions of the original image. Thus we need to apply only fixed size 3x3 feature detector masks to different levels of the pyramid [3]. This is equivalent to applying filters of increasing size to the original image.

The key issue in many dynamic scene analysis problems is that of Motion estimation. It has been an area of research over the past two decades. It has applications in a broad range of fields including video coding by motion compensation, data compression, autonomous navigation and object tracking. Motion estimation requires the computation of the Optical flow vectors which gives the time rate of change of the intensity values along the x and and y directions. There are two approaches for estimating optical flow fields. First one is feature based approach where correspondence is established between two frames to estimate the flow vectors. Second one is the spatio-temporal gradient based approach where motion vector of each pixel is estimated. Most of the gradient based approach assume that the intensity variation within a neighborhood should be kept linear. This limits the size of the vector to the range in which the intensity varies linearly. Moreover these algorithms yield significant error across motion boundary. In this thesis we proposed hierarchical algorithm where interlevel motion smoothness constraint is used rather than the local smoothness constraint, thereby alleviating the above problems. The multigrid technique is used in the algorithm which increases the speed of the algorithm [4].

Organization of the Thesis

Chapter2 describes the real-time image processing system(RTIPS) used for implementing the hierarchical algorithms. The system organization followed by the building of pyramid, using this system, is also discussed.
In Chapter3 we describe the implementation of the hierarchical algorithms for line extraction based on Hough Transform(HT) , Texture Discrimination using the energy map of the pyramid and Optical Flow Estimation on the RTIPS.
Chapter4 gives the results of the algorithms described in Chapter3.
Chapter5 concludes and brings to view some further scope of research in the related field.

Chapter 2

The Pyramidal Architecture for Real-Time image processing

The two configurations for parallel processing that are commonly used can be classified as:

(1) In which the processors can share **centralized global** information. But this configuration requires complex mutual exclusion and synchronization.

(2) In which the global information of the image data is **distributed** to all processors. Each processor can then work on its sub-image, making use of global information. This configuration requires that every modification of the global information is reported to all the processors in order to maintain overall consistency. These drawbacks of the above systems slows down the execution process and decreases the degree of

parallelism in both the systems.

2.1 The Pyramidal system

To increase the speed of the processor the volume of data to be processed is to be reduced . This approach is related to the concept of multiresolution. There is also a need for a system in which the inter-processor communication is very fast so that no overheads are involved while transferring the data. Thus we are looking for an architecture that achieves data reduction and can handle both local-distributed and global-centralised information(data). This architecture is known as the pyramidal architecture. In this architecture each higher level has *o*ne fourth of the data at the immediate lower level. The lower level has the local information (the intensity values of the pixels) of the image whereas the higher level has more global information (such as the averaged pixels information from the lower level). The pyramidal architecture thus is a useful compromise between the need for a parallel architecture and the hierarchical representation of the image at different resolutions.

2.1.1 The Pyramid model

The pyramidal model can be characterized by:

- a tessellation of the image plane;

- a set of(elementary) processors;
- a hierarchical communication network.

We shall only consider the pyramids with regular square tessellation.

2.1.2 Communication within the pyramid

The communication within the pyramid is made of two kinds of links.They are:

(1) Horizontal, i.e. intra-level links

This link is between the processors on the same level(k) of the pyramid. This link is represented by a relation between the pixels of the adjacent processors. let $\eta(P, Q)$ be the relation defined on Image I_k (k is the level) as follows:

$\eta(P.Q) \equiv$ Pixel P is a neighbor(brother) of Pixel Q i.e. P and Q belong to adjacent processors on the same level.

η is non-reflexive, symmetric and non-transitive.

The neighbors are defined according to their connectivity. The 4-connectivity and 8-connectivity based neighbors are shown in figure 2.1. (2) Vertical, i.e. inter-level links

This link is between the processors on the adjacent levels of the pyramid which is defined by the following relation between the pixels: Let $\Re$ be the relation defined on the pyramid by

$\Re(P, Q) \equiv$ P is a parent of Q.

	x y-1	
x-1 y	x y	x+1 y
	x y+1	

x-1 y-1	x y-1	x+1 y-1
x-1 y	x y	x+1 y
x-1 y+1	x y+1	x+1 y+1

Figure 2.1: The 4-connectivity and 8-connectivity based neighbors

That is pixel P is at a level exactly above Q. $\Re$ is non-reflexive, antisymmetric and non-transitive [3].

2.2 Types of pyramid

Depending on the arrangement of the processors, the pyramidal models are classified into two major types:

(1) Bin-Pyramid

The architecture is called a *Bin-Pyramid* if every node has two children. In order to achieve complete coverage of the image, we must alternate row-children and column-children. However it is very costly because it has $2^{2N+1} - 1$(N is the number of levels in the Pyramid) nodes and the number of levels is also large(2N+1).

(2) Quad-Pyramid

This is the most common architecture. In this case, every node(x,y,k) has four children (2x. 2y, k-1), (2x+1, 2y, k-1), (2x, 2y+1, k-1), (2x+1, 2y+1, k-1) and one parent (x/2, y/2. k+1) where the division is an integer division. In this architecture the number of nodes is $\frac{4}{3}.2^{2N}$ and the number of levels is N+1 which are much less as compared to those in Bin-Pyramid. The quad pyramid is shown in figure 2.2.

An overlapped Quad-Pyramid is obtained if we add the neighbors of four central children(using 8 connectivity). Thus for any node, there are four parent cells at the level just above and sixteen children at the level just below it. We have used this configuration for our system. The parents are $(\frac{x}{2}, \frac{y}{2}, k+1)$ $(\frac{x}{2}, \frac{y}{2}+2(y\ mod\ 2)-1, k+1)$ $(\frac{x}{2}+2(x\ mod\ 2)-1, \frac{y}{2}, k+1)$ $(\frac{x}{2}+2(x\ mod\ 2)-1, \frac{y}{2}+2(y\ mod\ 2)-1, k+1)$. The children are (u.v,k-1) where u=2x-1,···,2x+2 and v=2y-1,···.2y+2.

2.3 System Description

2.3.1 System topology

The topology of the system is shown in figure 2.3. The system comprises of 4 processors at the base and one at the apex in the form of a pyramid. The processor at the apex has the following functions

(1) receives the image data,

(2) preprocess it,

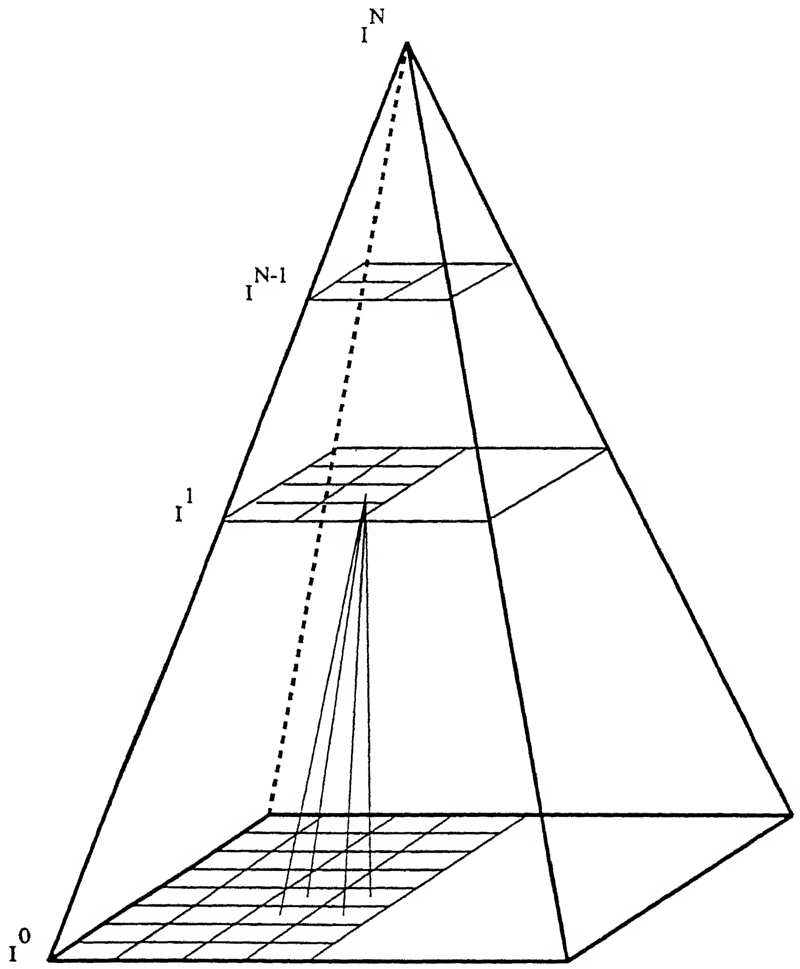


Figure 2.2: Quad-Pyramid : N+1 is the number of levels

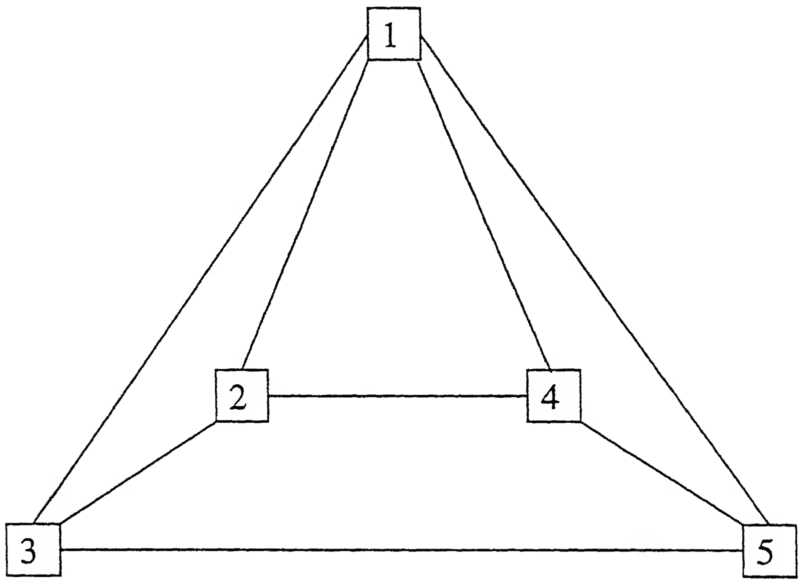


Figure 2.3: Topology of the system

(3) divides it into 4 subimages (each $\frac{M}{4}$x$\frac{N}{4}$) and

(4) passes the subimages to the processors at the base.

Each processor at the base builds a sub-pyramid by operating on the subimage. The processor at the apex finally combines the results from all the sub-pyramids.

2.3.2 System block diagram

The block diagram of the RTIPS that has been configured is shown in figure 2.4. The system consists of three parts. (1) Host computer (2) Frame grabber and (3) DSP cards.

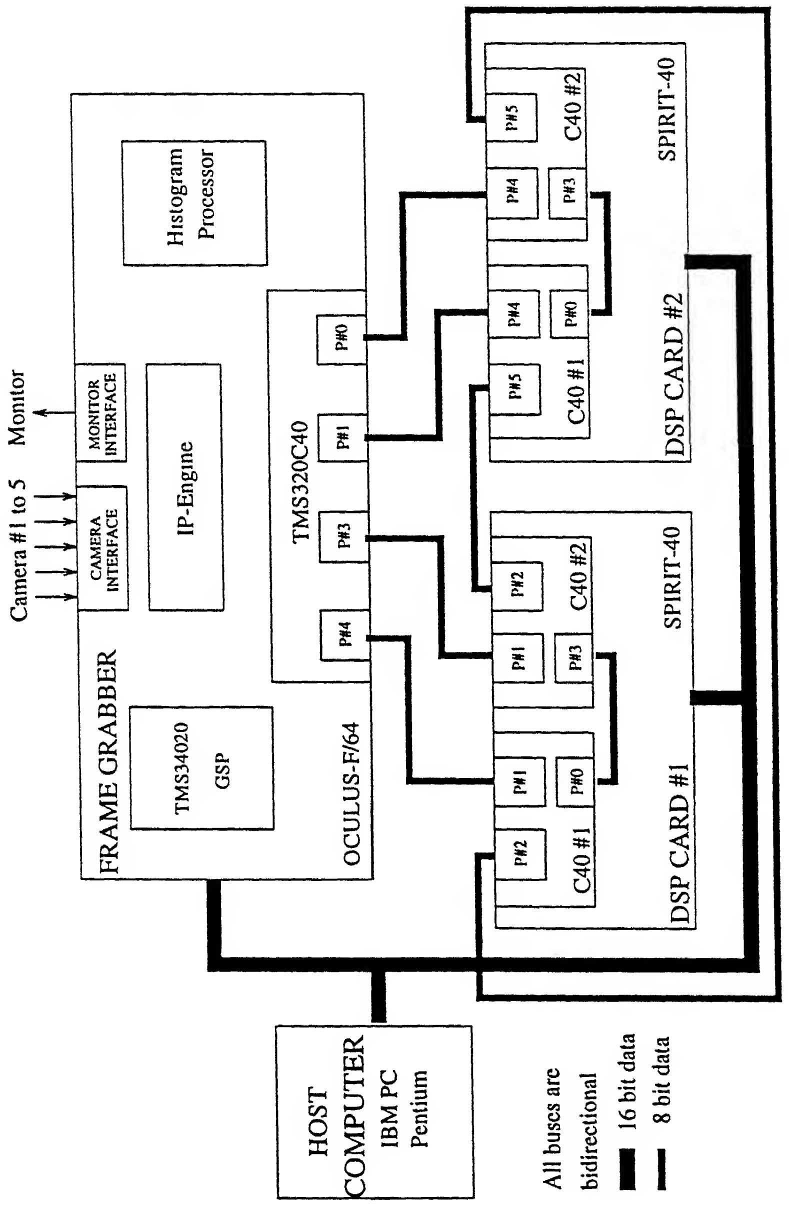


Figure 2.4: Block diagram of the system

- Host computer

  The host used is an IBM-PC, Pentium (100MHz). The functions of the host computer are (a) to download code to the processors in the frame grabber and the DSP cards and (b) to get any frame of interest for further analysis.

- Frame grabber

  The Oculus-F/64-DSP card (ISA bus compatible) made by Coreco Inc. is used to grab the video signals [5]. It can support various kinds of cameras. Up to four analog cameras, (one capable of accepting differential signal) and one digital camera(with 8/10 or 12 bit data path) can be connected to this frame grabber. It is capable of acquiring data at a rate of up to 40 million pixels/second. The frame grabber includes a TMS34020 graphic signal processor (GSP), TMS320C40 DSP, IP-Engine and a Real-time Histogram Processor. Functions of each processors on the frame grabber are as follows.

  1. TMS34020 GSP (40MIPS) :

  It is responsible for the acquisition control (up to 40 million pixels/second) and display with basic image processing functionality.

  2. TMS320C40 DSP (25MIPS, 100MFLOPS peak) :

  This DSP can be programmed for enhanced acquisition sampling rates as well as for powerful image processing functions. All computationally intensive programs are run on this processor(This is the processor which acts as the apex of the pyramidal structure). Besides this it gives the communication interface to other TMS320C40 DSPs on the DSP cards through its full duplex communication channels.

3. IP-Engine(40MIPS) :

This performs real-time image averaging, subtractions and logical operations. It is custom programmed

4. Histogram Processor :

This is a real-time histogram processor which produces histograms on image frames. Real-time histograms facilitate intensity analysis, object identification and object detection.

5. Memory ·

VRAM frame buffer 2MBytes, DRAM frame buffer 1MByte, GSP program memory 1MByte standard, expandable to 32 MBytes, Maximum VRAM memory expansion 16MBytes, Maximum DRAM memory expansion 32 MBytes, Overlay memory expansion 4 MBytes.

- DSP Cards

  The SPIRIT-40 cards(ISA bus compatible) made by Sonitech International, are used for computationally intensive image processing functions [6]. The key features of the SPIRIT-40 are:

  1.Spirit-40 card contains two TMS320C40 DSPs.

  2.Four independent 32-bit RAM banks (2 local/private, 2 global/dual-access). Each bank holds 256k x 32 of 0 wait-state SRAM, for a total of 1M x 32 of SRAM

3. Ten C40 communication ports (five from each C40) available, with 20 MBytes/sec. transfer rate.

4. Real-time operating system(RTOS) application programming interface support.

The processors TMS320C40 used in SPIRIT-40 offer the following features [7].

1. 40ns single-cycle instruction execution time.
2. 20MBytes/sec. transfer rate/communication port(there are 6 comm ports).
3. Six on-chip DMA controllers for concurrent I/O and CPU operations.
4. Two 1k x 32 bit single-cycle dual access on chip RAM blocks.
5. 128 x 32 bit instruction cache.

2.3.3 Application Development on the RTIPS

To develop programs on the RTIPS we need to program all the sub-systems, viz. the PC, the TMS320C40 DSPs available in the frame grabber and in the SPIRIT-40 card. So corresponding to every application, there should be executable codes for all the processors, all the DSP chips, the frame grabber components and the PC. It is upto the programmer to decide the interprocessor communication timings, data etc.

Normally for a real-time image processing application the flow chart would be as follows:

1. Develop executable codes for all the DSPs (one in frame grabber and two in SPIRIT-40).
2. Download executable codes (*.out files) to the respective DSPs.
3. Get the image frame from frame grabber.
4. Distribute the frame data among the DSP processors.
5. Do the computationally intensive operations in DSPs (provided the algorithm is parallelizable).
6. Once the computation is over, get the processed fragmented data from various DSPs and reassemble to get the complete processed image frame.
7. Transfer the processed data to the display.
8. Repeat the processes from 3 for the next frame data.

2.3.4 Speed performance of the System

Assuming frame frequency to be 25frames/second, the data is to be communicated and processed within 40ms. For a frame of resolution 512 x 512 the data to be processed is 256kwords, i.e., 1MBytes. Since the communication port has 20MBytes/sec. transfer rate, 1MBytes data can be transferred within 12.5ms. The processed data is to be send back to the apex (processor 1). So the total time required for inter-processor communication is 25ms. The remaining 15ms. of the frame time is available for processing, within which each processor can execute 375k (15ms/40ns) instructions. Most of the hierarchical algorithms for computer vision applications can be implemented within this limit (375k instructions).

2.4 Building The Pyramid

A hierarchy is defined as a set of layers, each of them corresponding to a given level of abstraction. Let I be an image of size 2^Nx2^N. In order to construct the pyramid the image is smoothened by a filter(in order to remove the high frequency components) and downsampled by 2. Thus the subsampling of the 2D signal combined with a smoothing process is done by a discrete convolution given by:

$$G[I](i,j) = \sum_{m=1}^{m=M} \sum_{n=1}^{n=M} w(m,n).I(2i+m-z, 2j+n-z) \qquad (2.1)$$

where

- M is the width of the convolution mask.
- z is a constant z=int($\frac{M+1}{2}$).
- (i,j) $\in$ [0,2^{N-1}-1]x[0,2^{N-1}-1]

Downsampling means that, in effect, the generating kernel doubles in size relative to the original image. Thus the band limit of pyramid levels decreases in octave [8]. W is usually a low pass filter.

The following constraints are proposed for the generating kernel [9]:

(a) Normalization

$$\sum_{m=1}^{m=M} \sum_{n=1}^{n=M} w(m,n) = 1 \tag{2.2}$$

The property domain is thus kept at its previous value(no gain, attenuation or amplification).

(b) Symmetry

$$w(m,n) = w(M+1-m,n) = w(m,M+1-n) = w(M+1-m,M+1-n) \tag{2.3}$$

for all m,n

This constraint is a consequence of the absence of an absolute coordinate system in an image.

(c) Unimodality

$$0 \leq w(m,n) \leq w(p,q) \; for \; m \leq p < \frac{M}{2} \;\; and \;\; n \leq q < \frac{M}{2} \tag{2.4}$$

This constraint makes sure that the larger weights will be at the center of the mask. Thus it contributes to preserving the continuity of the signal.

(d) Equal contribution to the next level

In order to avoid distortion of the signal, all pixels of I must contribute the same total weight to each pixel of G[I]. This constraint is automatically satisfied for the even kernels:

$$\sum_{i=0}^{i=\frac{M}{2}-1} \sum_{j=0}^{j=\frac{M}{2}-1} w(2i+m, 2j+n) = \frac{1}{4} \; for \; m, n = 1, 2 \tag{2 5}$$

(e) Separability

$$W = w_1^t.w_2 \; i.e \; w(m,n) = w(m).w(n) \tag{2.6}$$

In particular case w_1=w_2=w is generally assumed. W is valid for building scale-space representation of an image only if the number of local extrema in the convolved image does not exceed the number of local extrema in the original image. The Gaussian kernel has been proved to be the only function satisfying this constraint [3]. For very high speed computational purpose the Gaussian kernel can be approximated by a set of binomial masks. The generic kernel w_2 is $\frac{1}{4}(1,2,1)$. By successive multiplication of this simple generic mask, we get a set of odd masks. For instance
w_4 is $\frac{1}{16}(1.4,6,4,1)$.

The general formula for the generic kernel is
$w_k = \frac{1}{2^k}(\binom{0}{k}\binom{1}{k}\cdots\binom{\frac{k}{2}+1}{k}\cdots\binom{1}{k}\binom{0}{k})$.

Figure 4.1 shows the Gaussian pyramid of the Lenna image. The generic kernel used is $w_3 = \frac{1}{8}(1,3,3,1)$. Using equation 2.6 we obtain the filter which is :

$$\frac{1}{64}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}$$

Chapter 3

Description of Hierarchical Algorithms

3.1 Straight line extraction

The problem of line detection is one of establishing meaningful groups of edge points which lie along straight lines. A classical way to detect edge points which satisfy a collinearity constraint is the Hough transform (HT). The HT has also been applied to the detection of edge point groups which lie along other analytic and non-analytic curves [2].

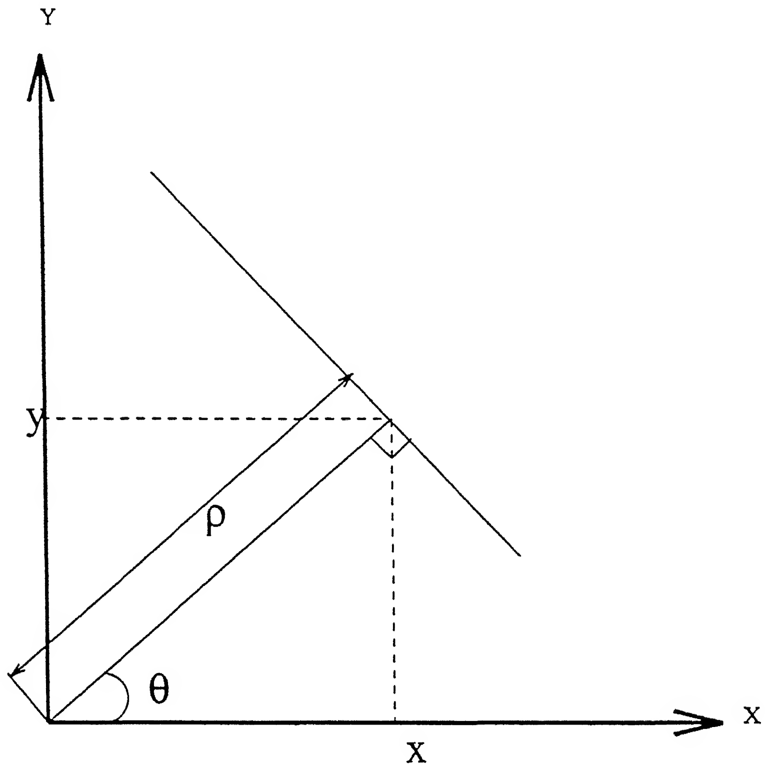


Figure 3.1: ρ, θ representation of straight lines

3.1.1 Pyramid Hough Transform for line extraction

Basically the Hough transform operates by mapping each edge pixel(x,y) into Hough space (ρ, θ). Where θ is the slope of the detected edge and ρ is the perpendicular distance from origin to the line through the edge point, having slope θ as shown in figure 3.1. From the figure 3.1 it is clear that

$$\rho = x\, cos\theta + y\, sin\theta \tag{3.1}$$

Under the above transformation, a set of collinear edge points map into the same position in Hough space. Thus a compact description of straight lines is obtained.

Signal to noise ratio of the Hough transform is higher for smaller blocks [3] Making use of this fact, each layer in the pyramid is split into a number of subimages At the bottom level of the pyramid short line segments are detected by applying a Hough transform to small subimages. We start from a local description of contours in terms of short line segments which is obtained by applying HTs to a large number of overlapping subimages which cover the original image. The method proceeds from this low-level description in a bottom up, hierarchical manner grouping line segments in local neighborhoods into longer lines. The grouping mechanism used in each parent is based on a HT type algorithm. Line segments which are found to have *l*ocal supporti.e which form strong groups propagate up the hierarchy and take part in grouping at higher levels while lines with no local support at a particular level terminate (there are a number of ways in which local support can be defined). Hence a hierarchical description of the line segments is obtained since length determines approximately the level to which a segment propagates. The method maintains an *i*nsensitivity to missing data points and is very efficient [2].

3.1.2 Line Segment Grouping

A group is a set of feature points which satisfy some constraint. In the case of collinearity we can express this constraint in an analytic form. For example edge points, $\overline{X} = (x, y, \phi)$, which lie along a line satisfy

$$\rho_0 = x\, cos\theta_0 + y\, sin\theta_0 \ , \ \phi = \theta_0 \tag{3.2}$$

for a particular value of (ρ_0, θ_0), where ρ_0 is the distance from the origin to the line(edge) along a direction normal to line and θ_0 is the angle of this normal with respect to the x axis . A collinear group could be thought of as a set with attributes (ρ_0, θ_0).

$$G(\rho_0, \theta_0) = \{\overline{X}_1 \ , \ \cdots \ , \ \overline{X}_N \ \mid \rho_0, \theta_0\} \tag{3.3}$$

The constraint given in equation 3.2 will not be satisfied exactly for a number of reasons:

(1) finite resolution of the imaging system,

(2) roughness in the actual lines, and

(3) edge points are defined on a discrete grid.

The discrete representation implies that the exact constraint will be violated in respect of both ϕ and ρ and a more reasonable approximate constraint for determining collinearity is

$$\rho_0 - \frac{\triangle\rho}{2} \leq x\, cos\theta_0 \ + \ y\, sin\theta_0 \leq \rho_0 + \frac{\triangle\rho}{2} \tag{3.4}$$

and

$$\theta_0 - \frac{\triangle\theta}{2} \leq \phi \leq \theta_0 + \frac{\triangle\theta}{2} \tag{3.5}$$

Here $\triangle\rho$ is fixed, say 1.5 pixels.

$\triangle\theta$ is function of subimage size. $\triangle\theta$ is inversely proportional to the size of the window.

The votes(for grouping) are accumulated according to the constraint of equation 3.4 and 3.5. That is we define a sampling of the parameter space and for each feature point we accumulate votes for each set of parameters which satisfies the constraint. This is similar to the quantization method, except that the sample spacings (denoted here by $\rho_\triangle$ and $\theta_\triangle$) has been decoupled from the constraints as expressed by $\triangle\rho$ and $\triangle\theta$. The sampling rate should be high enough to distinguish all lines which could be present in the discrete subimage. The number of orientations which can be distinguished depends approximately linearly on the image size. The θ sampling rate chosen is $\theta_\triangle = \frac{\pi}{4L}$
where L is the length of the central portion(retina) of the subimage as shown in Figure 3.2 . The particular samples chosen were

$$\theta_k = \frac{\pi k}{4L} \; k = 0 \cdots 4L - 1 \tag{3.6}$$

The ρ sampling rate is $\rho_\triangle$=0.5.
In this work we have used the following range for the parameters:

$$-\frac{L}{2} \leq \rho \leq \frac{L}{2} \tag{3.7}$$

and

$$0 \leq \theta < \pi \tag{3.8}$$

We can weaken the proximity constraint by using larger local neighborhoods and fewer levels in the pyramid. However, it seems that the actual requirements are not fixed and should be a function of the complexity of the scene. In simple scenes when

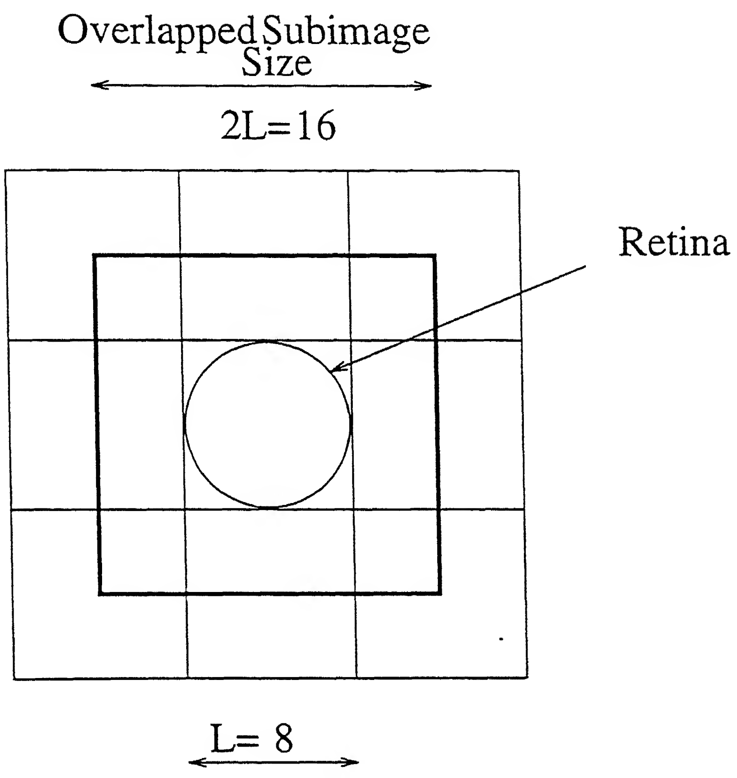


Figure 3.2: Overlapped subimage used in Line Extraction

the feature density is low stringent proximity constraints are neither necessary nor desirable As the scene complexity increases the imposition of proximity constraints becomes more important.

3.1.3 Algorithm

A conventional pyramid is employed with (P+1)levels. In our case P=4 At each level l, the image is divided into 2^{P-l} x 2^{P-l} subimages, where P is the top level of the pyramid. representing the complete image. The algorithm proceeds in a bottom up manner starting with a description of the image in terms of line segments detected in the subimages at the bottom level (level 0). These line segments are detected by applying a conventional HT algorithm to the set of edge points found by any edge detection method and thresholding. Since the HT is calculated in a small subimage the problem of differentiating between a true line and a random alignment of points is reduced. The lines are reconstructed using the end points in the groups and so even if the intermediate points are occluded their absence is taken care of to certain extent. The method is also relatively efficient since only a small accumulator array is needed to represent all possible lines which pass through a subimage.

(a) Processor 1 (Refer figure 2.3) detects the edge pixels of the base. Let the gradient magnitude and direction be m(x,y) and $\theta(x,y)$ respectively. (Refer [10] for edge detection.)

(b) Each of the 4 processors receives the x, y, m, ρ values corresponding to the $\frac{M}{4}$ x $\frac{N}{4}$ block from processor 1.

(c) At each processor the image is further divided into sub-blocks. Thus the bottom level of the pyramid consists of subimages, regularly arranged on the retina and the initial description identified local line segments from these subimages. The subimage size determines several properties of the line segment detection. Small subimages can be useful in complex scenes since in a local view leads to a HT which is straightforward to interpret and has little chance of forming spurious groupings particularly if the detection threshold is high. Thus good results can be obtained at low computational cost by choosing smaller sub-blocks. However, making the subimage too small can result in an algorithm which is sensitive to missing edge points. We have chosen the overlapped subimage size as 16x16. Each processor then finds best threshold(from the histogram of each subimage) to apply to the magnitude of its edge pixels in these sub-blocks.

(d) Each processor finds best partition of (ρ, θ) values in these sub-blocks. At the lowest level a group consists of a set with one member and each subimage will contain a list of groups which represent the line segments passing through it. The parameter space limits imposed on the HT determine which lines passing through the subimage will have detectable peaks. Once the HT is accumulated the results must be interpreted as evidence for line segments in the image. Simple thresholding does not provide adequate performance. Our results have been obtained by accumulating the

HT, finding the maximum, removing points from the edge list which contribute to this maximum, and reaccumulating the HT to find further peaks. Obviously, for a HT applied to the full image such a method would be inefficient, since a separate HT must be accumulated for each line found in the complete image. However, given a small subimage the number of lines present is small(usually 1 or 2 for 8x8 subimages) and therefore the HT is accumulated for small number of times. Also, a large number of the edge points are usually associated with single line segments and therefore subsequent accumulations use a much smaller set of edge points. Under these conditions the method becomes quite efficient. The process of repeated accumulation is terminated when the peak found falls below a preset threshold. The threshold is usually chosen to be 50% of the overlapped subimage size. In our case, since the subimage size is 16x16, the threshold is 8.

(e) For each processor the parent at level 1 gets the information of groups from the 4 children. At level 1, more global groups are formed by allowing each segment detected in the siblings(from level 0) to vote for lines in the parent accumulator. If each sibling line segment gives a single vote for the lines which it could be a part of, parameter values with two or more votes represent possible groups and these are examined on a best first basis. For each peak found a new group is determined as the union of all contributing groups and is given the parameters defined by the position of the peak in the parent HT. Hence each group above the lowest level in the tree has a set of attributes determined from the local HT and a list of subimages from the lowest level which represent the extent of the line. Several conditions are imposed to determine whether this new group is valid. If the size of the new group is larger than any of the contributors i.e, if we label contributors $G_i(\rho_i, \theta_i)$ and call the newly formed group, $G'(\rho_0, \theta_0)$ we must have

$$G'(\rho_0, \theta_0) \supset G_i(\rho_i, \theta_i) \ and \ G'(\rho_0, \theta_0) \neq G_i(\rho_i, \theta_i) \ \ for \ all \ i \qquad (3.9)$$

and if the contributors come from connected sibling subimages then the new group is valid and all the contributors are removed from the line segment lists in their originating subimages. If equation 3.9 is not true then we must have $G'(\rho_0, \theta_0) = G_i(\rho_i, \theta_i)$ for some i. The new group is not formed in this instance and all contributing segments except one for which $G'(\rho_0, \theta_0) = G_i(\rho_i, \theta_i)$ are removed from their originating subimages. In this case there is no new evidence to support the existence of a more global underlying line and the line segment should not participate in grouping at higher levels. Single votes are ignored and therefore they remain on the line segment list in their originating subimage. In this way line segments which have a more global significance are propagated up the hierarchy on the basis of local support, while line segments with no local support do not participate in grouping at higher levels. The grouping process at each parent is thus very simple, since only a small number of line segments participate. The number of line segments which exist at high levels is small owing to the rarity of long lines.

Since the points are represented with respect to a local origin x_s, y_s and the origin of the parent subimage is x_p, y_p, the appropriate parameter space curve is given by

$$\rho = (\rho_0 \ cos\theta_0 \ - \ (x_p - x_s) \ )cos\theta + (\rho_0 \ sin\theta_0 \ - \ (y_p - y_s) \ )sin\theta \qquad (3.10)$$

Letting $\triangle x = x_p - x_s$ and $\triangle y = y_p - y_s$, with some manipulation we obtain

$$\rho = \rho_0 \, cos(\theta - \theta_0) - \triangle x \, cos\theta - \triangle y \, sin\theta \quad (3\ 11)$$

(f) Step(e) is repeated till a single parent at the apex has a collection of all the groups from all the subimages of each of the 4 processors

(g) The processor 1 finally combines the groups from the 4 processors to form a group which is the largest and strongest group and represents largest straight edges in the image.

Compared with the classic Hough transform this algorithm achieves economy of storage space and processing time, because it works with only clusters and never constructs the entire Hough space.

3.2 Texture Discrimination

Texture is made up of individual elements called *t*extons whose distribution forms the texture pattern at a macroscopic level. Texture segmentation is defined as the division of the image into *h*omogeneous regions where local texture properties are approximately invariant. Despite many attempts there is still no appropriate model for *h*omogeneous textures [1]. Texture homogeneity is defined only with respect to our visual perception. A region is said to have a homogeneous texture if it is preattentively perceived as being homogeneous by a human observer. The goal of texture discrimination is to find a minimum number of measurements that can discriminate textures that are perceived to be different These measurements should also remain approximately constant in a region where the texture is considered to be homogeneous. The orientation of the texture elements and their frequency contents seem to be important clues for discrimination

To design an effective algorithm for texture segmentation, it is essential to find a set of texture features with good discrimination power. Most texture features are generally obtained from the application of a local operator, statistical analysis, or measurement in a transformed domain. Common features are estimated from co-occurrence matrices. Law's texture energy measures. Fourier transform domains. Markov random field models, local linear transforms, etc [11]. While in most early work textures are analysed in a single resolution, more recently, attention has been focused on *m*ultiresolution and *m*ultichannel texture analysis.

Textures can often be described in terms of a two-level pattern At low, or *m*icroscopic level, this consists of the patterns formed by individual texture grain elements, and at a higher, *m*acroscopic level it consists of repetitive patterns formed by the distribution of these grain elements. Thus two scale parameters characterize these textures: the grain size, and the repetition distance. This emphasizes the role of multiscale analysis of texture for their discrimination. To analyze texture. the image is convolved with a set of feature detectors. The outputs are then combined to produce a unique texture response using energy measure as the criteria for discrimination [3]. Even simple texture contains features at more than one resolution. Therefore to extract features with different frequency contents we need to apply filter of different sizes(i e different band widths). This implies that we must use more than one size for masks Thus the disadvantage of this approach is that we have to choose the size of the mask. In hierarchical approach however, this can be taken care of as we need to apply only a set of 3x3 masks to the image at all resolution levels.The image pyramid contains low resolution versions of the original image at higher levels. Therefore when we convolve the higher level with the same 3x3 filter. effectively more region of the image is masked by the filter as compared with that when filtering is done at the lower levels. Thus effectively the filter size is increased when we apply the 3x3 mask to the image at higher level. Thus the problem of deciding the filter size is taken care of in this approach.

3.2.1 Algorithm

Here we present the method of building a texture pyramid based on multiscale feature analysis. The first part(steps 1-3) of the algorithm consists of building the *f*eature pyramid.

1 Compute intensity pyramid from level 1 to N(number of levels). Refer section 2.4 for building the pyramid.

2 Compute the feature response for each level.

In this step we convolve a set of 3x3 masks with the images at all the levels

$$f_{kj}(P) = M_j(P)\ G_k(P) + \sum_{Q \in Brothers(P)} M_j(Q).G_k(Q) \qquad for 1 \leq j \leq J \tag{3.12}$$

where J=8, the number of masks(feature detectors).

Here G is the Gaussian pyramid image.

k is the level of the pyramid.

M_j is the mask or the feature detector(j=1 to 8).

"*Q is brother of P* "implies that pixel Q is a neighbor of P.

The feature detectors used are

-1-2-1	-1 0 1	1-21	1 0 -1
0 0 0	-2 0 2	0 00	2 0 2
1 2 1	-1 0 1	1-21	1 0 -1

-12-1	-1 -2 -1	1 0 -1	1 -2 1
-24-2	2 4 2	0 0 0	2 4 2
-12-1	-1 -2 -1	-1 0 1	1 -2 1

These feature detectors are set of masks which are the discrete versions of the basis functions defined by Laws [3]. These masks compute local differences in various simple spatial configurations. The energy measure thus represents the distribution of edges and other local features in the image

3. Compute the texture pyramid for each level

$$F_k(P) = \sum_{j=1}^{j=J} |f_{kj}(P)| \tag{3.13}$$

Add the responses from all the masks at a given level to give the feature response of that level. The feature response from all the levels gives the feature response pyramid. Apply threshold to $F_k(P)$

$$F_k(P) = \begin{cases} 1 & if F_k(P) \geq s_k \\ 0 & else \end{cases} \tag{3.14}$$

s_k is the threshold obtained from the histogram of the summed image.

Of course, one can summarize the feature response by taking the sum itself. But, since thresholding produces binary images. no normalization is needed

We now consider the next part of the algorithm i.e. building the energy pyramid E for the texture pyramid F obtained from the above steps.

4. Compute the texture energy for each level

$$E_k(P) = \sum_{l=1}^{k} E_{kl}[F](P) \tag{3.15}$$

where $E_{kl}[F](P)$ is the energy obtained in point P(x,y) at level k, from signal at level l. $E_{kl}[F](P)$, the texture energy, is the sum of the energy of the texture pyramid at that scale(l).Since the texture pyramid is composed of binary values, the energy $E_{kl}[F](P)$ is simply the amount of texture of scale l in the receptive field of cell P.

5 To get the final output we project and sum all the levels of the energy pyramid (from a given level H) i.e. the energy responses are averaged.

$$T_H(P) = E_H(P) \tag{3.16}$$

$$T_k(P) = E_k(P) + T_{k-1}(Q) \tag{3.17}$$

where Q is the parent of P(i.e Q is at higher level than P).

H is the range over which the energy response is averaged. Thus T.E.M is a function of H. We recursively solve equations 3.16 and 3.17 till we reach level 1(at the bottom level k=1) where T_1 is evaluated. T_1 is the texture energy map(T.E.M.). T.E.M. indicates the amount of texture in the neighborhood of each pixel. T.E.M. is helpful for general image analysis and provides an initial decomposition of the image into its main components which is used for texture segmentation.

The projection operator used is $\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

which is two-dimensional bilinear interpolation [12].

The result of the Texture Discrimination algorithm is given in chapter 4.

3.3 Optical Flow Estimation

A visual motion perception plays important roles in many areas, such as perception of 3D structures, tracking of moving objects, and segregation of visual field into meaningful segments. The perception of visual motion depends on the intensity variation with time throughout the visual field. The temporal shift of observable gray value structure forms the flow field of visual patterns on the image plane. This field, which is called an optical flow, specifies the instantaneous velocity of the corresponding image component. In general, there are two approaches in estimating the optical flow fields. The first is the *f*eature based approach and the second one is the spatio-temporal gradient based approach.

Gradient based optical flow estimation algorithm makes use of the spatio-temporal gradient relationships in the local neighborhood within which the displaced point resides. However, the spatio-temporal gradient relationship only estimates the motion in the direction of gradient. In order to alleviate this problem, Horn and Schunck [13] explicitly formulated a local smoothness constraint on the motion vector. More specifically, by assuming that the flow field varies smoothly with the image coordinates x and y(i.e. local smoothness is assumed), they showed that a complete motion vector could be found. However, it is noted that the local smoothness constraint yields a significant estimation error across the motion boundaries.

The success of the gradient based approach is dependent on the image characteristics of the neighborhood. Computation of optical flow uses the optical flow constraint (equation (3.18)) together with an additional assumption. The assumptions include one of the following [14]:

(a) optical flow is smooth and neighboring points have similar velocities,

(b) optical flow is constant over an entire segment of the image,

(c) optical flow is the result of restricted motion, for example, planar motion.

Most gradient based algorithm such as Horn and Schunck's assume that the intensity variation in the local region should be kept linear, since the gradient is essentially a linear approximation of the local image characteristics. Thus the maximum size of the motion vector is determined by the range in which the intensity varies linearly. In the other words, the image characteristics affect the range of the motion vector that can be reliably estimated by the gradient based approach. To cope with this problem, both Terzopoulos [15] and Glazer [16] applied the pyramid structure and multigrid method to Horn and Schunk's algorithm. The basic idea of using a pyramid structure is that the relatively large motion vectors can be estimated at the coarse level by allowing the intensity variation linearly over the large areas. However the algorithms employ the same motion smoothness constraints as in Horn and Schunck's and as a result both algorithms result in the smoothed optical flow fields and considerable errors especially in the motion boundary regions.

In this section we propose an optical flow estimation algorithm based on spatio-temporal gradients using the Gaussian image pyramid. The Gaussian pyramid is obtained by repetitively applying Gaussian filter and downsampling the image as discussed in section 2.4. Since the Gaussian pyramid has the smoothed versions of the original image the motion vector at each pyramid level is also a low pass filtered version of the motion vector at the finest level. Thus the interlevel motion smoothness constraint is introduced in the motion estimation. The overall smoothing effect in the flow field is reduced as compared to the methods based on the conventional spatial smoothness constraint.The employment of the pyramid allows us to estimate large flow vector and to adapt easily the multigrid technique to improve the convergence speed of the algorithm.

The flow field at each pyramid level varies smoothly as the level changes. Therefore by using the interpolation and averaging operators to establish the appropriate relationship between two adjacent pyramid levels, we can formulate the interlevel motion smoothness constraint to minimize the variation of flow across the pyramid levels.

3.3.1 Formulation of interlevel motion smoothness constraints.

The optical flow equation for the motion vector U(u,v) is

$$I_x u + I_y v + I_t = 0 \tag{3.18}$$

where I_x and I_y are spatial gradients along x and y directions and I_t is the temporal gradient defined as follows

$I_x = \frac{\partial I(x,y,t)}{\partial x}$, $I_y = \frac{\partial I(x,y,t)}{\partial y}$ and $I_t = \frac{\partial I(x,y,t)}{\partial t}$

I_x and I_y are obtained by applying edge detector operators to the image and I_t is obtained by taking the time difference between two consecutive image frames(assuming time interval=1).

The averaging P_k^{k+1} and the interpolation P_{k+1}^k operators used are

$$P_k^{k+1} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad P_{k+1}^k = \frac{1}{4}\left]\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}\right[$$

The additional constraint required for solving equation 3.18 is given by interlevel motion smoothness constraint which requires minimization of C(u^k, v^k) which is defined as

$$C(u^k, v^k) = \sqrt{(u^{proj} - u^k)^2 + (v^{proj} - v^k)^2} \tag{3.19}$$

Then the estimation of the flow vector $U^k(u^k, v^k)$ is equivalent to the minimization of

$$E(u^k, v^k) = \int\int [(I_x^k u^k + I_y^k v^k + I_t^k)^2 + \alpha\{C(u^k, v^k)\}^2] dx dy \tag{3.20}$$

The second term in the equation 3.20 is the deviation from the smoothness constraint. α is the Lagrange multiplier constant(0.6 in our case).
Making the partial derivative of E(u^k,v^k) with respect to u^k and v^k equal 0 gives the following Euler-Lagrange equations

$$\{\alpha^2 + (I_x^k)^2\}u^k + I_x^k I_y^k v^k = \alpha^2 u^{proj} - I_x^k I_t^k \tag{3.21}$$

$$\{\alpha^2 + (I_y^k)^2\}v^k + I_x^k I_y^k u^k = \alpha^2 v^{proj} - I_y^k I_t^k \tag{3.22}$$

Using the Jacobi relaxation technique [17] for solving equation 3.21 and 3.22, we have the following iterative equations

$$u^k = u^{proj} - I_x^k \frac{P}{D} \tag{3.23}$$

$$v^k = v^{proj} - I_y^k \frac{P}{D} \tag{3.24}$$

where,

$$P = I_x^k u^{proj} + I_y^k v^{proj} + I_t^k \tag{3.25}$$

$$D = \alpha^2 + (I_x^k)^2 + (I_y^k)^2 \tag{3.26}$$

Steps in relaxation technique

A basic relaxation consists of the following steps:

(1) Updating the finer level k (using equation 3.23 and 3.24) employing the projected flow vectors from level k+1.

(2) Fine to coarse(i.e. level k to level k+1) projection of the updated values.

(3) Updating at level k+1 (using equation 3.23 and 3.24) employing the projected flow vectors from level k.

(4) The coarse to fine(i.e. level k+1 to level k) projection of the updated values.

3.3.2 Multigrid method in relaxation.

One of the problem encountered with the relaxation method is its slow convergence behavior. The relaxation methods are known to be very efficient in smoothing errors [12]. After a few iterations on a given grid the high frequency error is considerably reduced while the low frequency error remains. The low frequency error is responsible for the slow convergence. In order to speed up the convergence, the multigrid method is used. When the convergence slows down we can approximate the remaining problem on the coarser grid(next higher level). The remaining error is at a higher frequency on the coarse grid, hence further relaxation at this level can reduce it effectively. When convergence is attained at the coarse level, the solution can be interpolated back to the finer grid(next lower level). Thus approximate solutions are sent down to the finer level after they have converged. When convergence slows at finer level they are sent up the pyramid for coarser processing. Different grid levels solve the problem in different spatial frequency bands[16]. Multigrid methods can converge in essentially O(N) number of operations, where N is the number of nodes in the grid. This can be compared to the complexities of $O(N^3)$ operations for the solution by standard(single level) relaxation [15]. Therefore, in image analysis applications, where N tends to be very large(order $10^4 - 10^6$, or more), multigrid method offers potentially dramatic increases in efficiency over standard relaxation methods.

3.3.3 Algorithm

```
$u^{proj} = v^{proj} = 0$
k=1                                              ;start at finest level
Until $U^1$ has converged Do
Begin
$u^k = u^{proj} - I_x^k \frac{P}{D}$                             ;do relaxation sweep
$v^k = v^{proj} - I_y^k \frac{P}{D}$
$u^{proj} = P_k^{k+1}(u^k)$                            ;project to next high
$v^{proj} = P_k^{k+1}(v^k)$                            level (average)
$k \leftarrow k + 1$                                     ;come up
$u^k = u^{proj} - I_x^k \frac{P}{D}$
$v^k = v^{proj} - I_y^k \frac{P}{D}$
$u^{proj} = P_k^{k-1}(u^k)$                            ;project down i.e.interpolate
$v^{proj} = P_k^{k-1}(v^k)$
$k \leftarrow k - 1$                                     ;come down(one relaxation over)
If( $U^k$ has converged ) Then                   ;if solution converges at
If $k > 1$ Then                                  coarse level then project
$k \leftarrow k - 1$                                     the solution to fine level
$U^k \leftarrow U^k + P_{k+1}^k(U^{k+1} - P_k^{k+1}U^k)$            and add correction
End If

Else If (slow convergence) Then                  ;if slow convergence at
If $k < M$ Then                                  fine level, then project up
$k \leftarrow k + 1$                                     for coarse processing and
$U^k \leftarrow P_{k-1}^k U^{k-1}$                              set the error threshold
End If                                           for next level
End If
End.
```

The error at level k is given by norm of

$$\{\alpha^2 + (I_x^k)^2\}u^k + I_x^k I_y^k v^k - \alpha^2 u^{proj} + I_x^k I_t^k +$$
$$\{\alpha^2 + (I_y^k)^2\}v^k + I_x^k I_y^k u^k - \alpha^2 v^{proj} + I_y^k I_t^k$$

Convergence is measured using L_2 norm of the error residual. The error must be below threshold. The threshold at the finest level(level 1) is supplied by the user. Threshold for intermediate levels is set when we project the solution up, which is 0.3 times the error at the finer level. Convergence is slow when the ratio of errors at a given level in the consecutive iteration is greater than 0.6. Chapter4 gives the results of the discussed algorithm applied to an image of a square block increasing in size in all directions and an image of a circle(ball) moving towards right.

Chapter 4

Results

This chapter includes the results of the algorithms discussd in chapter3. The system topology has been simulated by using only two processors in a SPIRIT-40 board. For implementing these algorithms we have used P.C.(Pentium) as the main interface between the user and the processors. The P.C. downloads the code and the data(half of the image data to each DSP) to the processors.
The straight line extraction algorithm was applied to the image of a cube(image size was 256x256). The overlapped subimage size chosen was 16x16 with the central retina size of 8x8. The grouping was continued till the 5^{th} level. The two processors were used in parallel by dividing the image into 2 equal parts and building the pyramids in parallel. The result (bottom one in figure 4.2) shows that the straight lines are extracted. These lines are sufficient to identify it as a cube.

The texture discrimination was applied to a 256x256 synthetic texture image and the T.E.M. was obtained as shown in figure 4.3. The T.E.M. clearly distinguishes the two textured regions.

The Optical flow estimation algorithm was applied to 128x128 image of a square decreasing in size between frames 1 and 2. The square block was of size 60x60 in the first frame and it was reduced to 40x40 in the second frame. The optical flow vectors are drawn with the scaling factor of 0.09 as shown in figure 4.6. These vectors clearly show the direction of motion of individual pixels. The frames were then interchanged and the resulting flow vectors obtained are shown in figure 4.7 with the scaling factor of 0.02. This algorithm was also applied to the image of a ball moving towards right. The radius of the ball was 24. The center of the ball was at (40,40) in the first frame and it was shifted to (40,60) in the second frame. The flow vectors are shown in figure 4.10 with scaling factor of 0.02. Except for the edges(since the edges are not well defined in the case of the circle), this result also gives the direction of motion of the ball.

The following table shows the CPU timings of the algorithm implemented on the HP UNIX system and the RTIPS.

Table 1

Algorithm	*HP workstation*	*RTIPS*
Straight Line Extraction	1.5min	0.5s
Texture Discrimination	2.7min	0.75s
Optical Flow Estimation	3min	1s

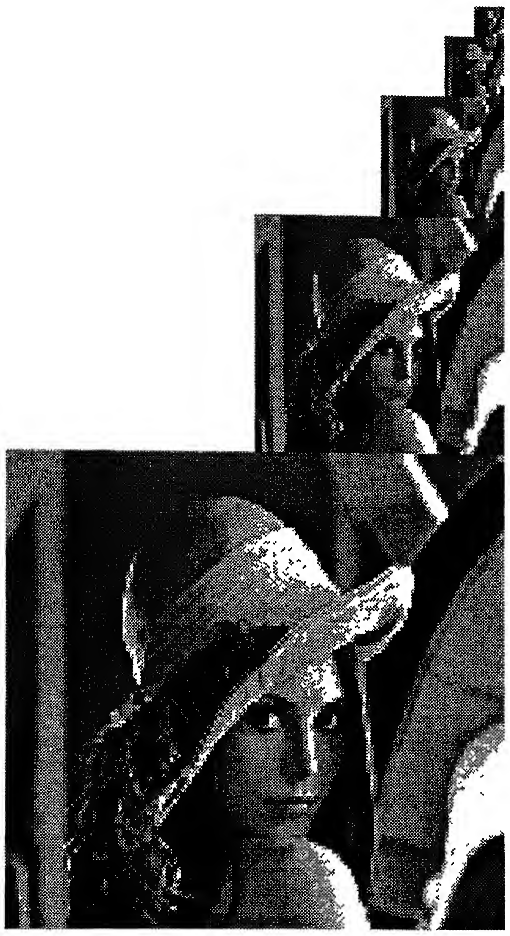


Figure 4 1: Gaussian Pyramıd of Lenna Image

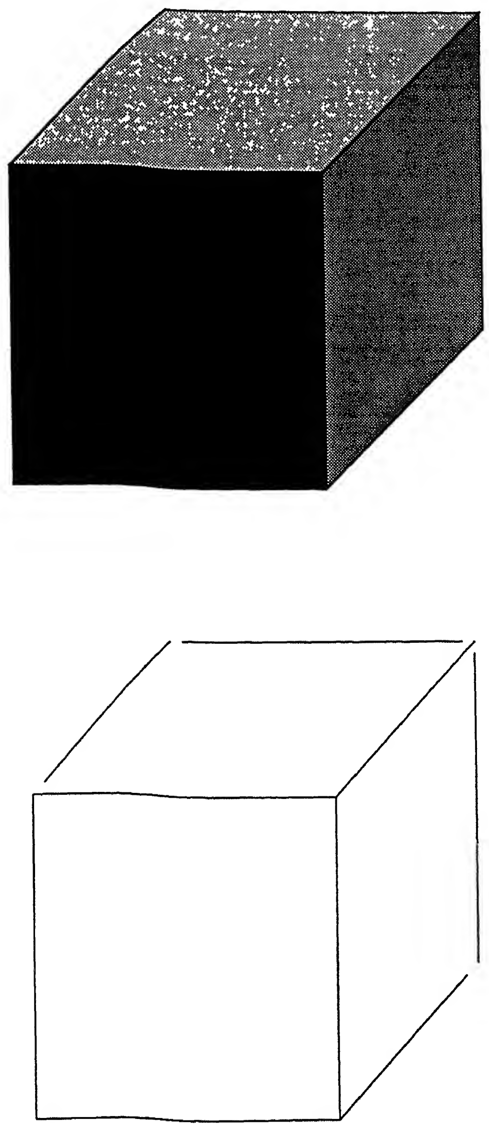
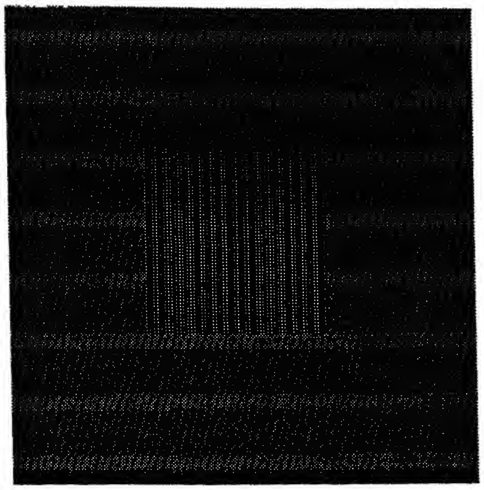


Figure 4.2: Result of Line Extraction applied on a cube



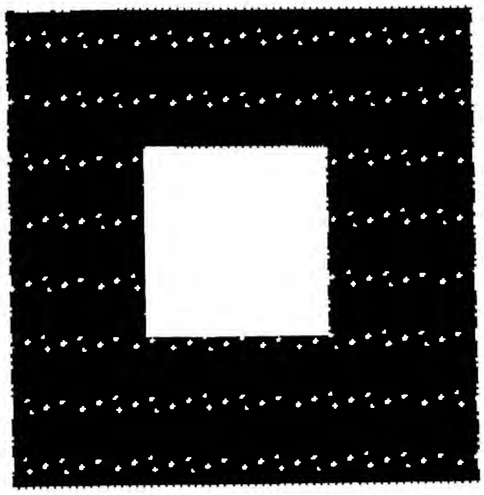


Figure 4 3: Texture Discrimination : Top shows the original texture image and bottom one is its T.E.M.

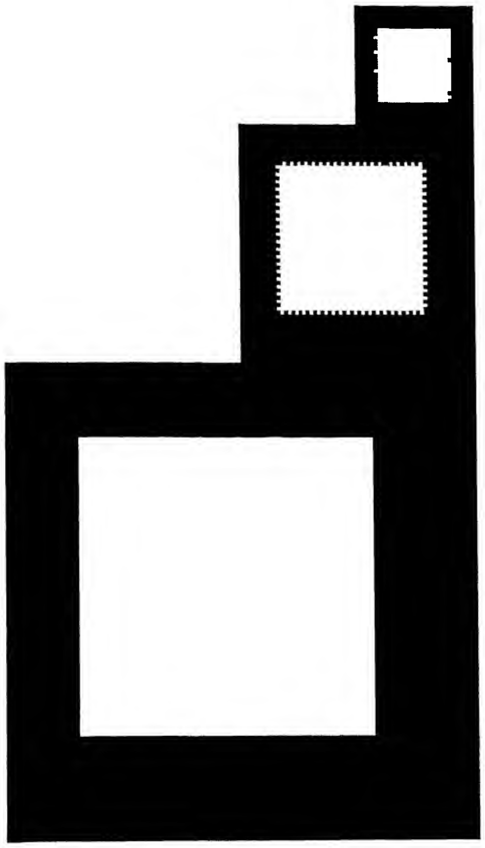


Figure 4.4: Gaussian Pyramid of First Image Frame

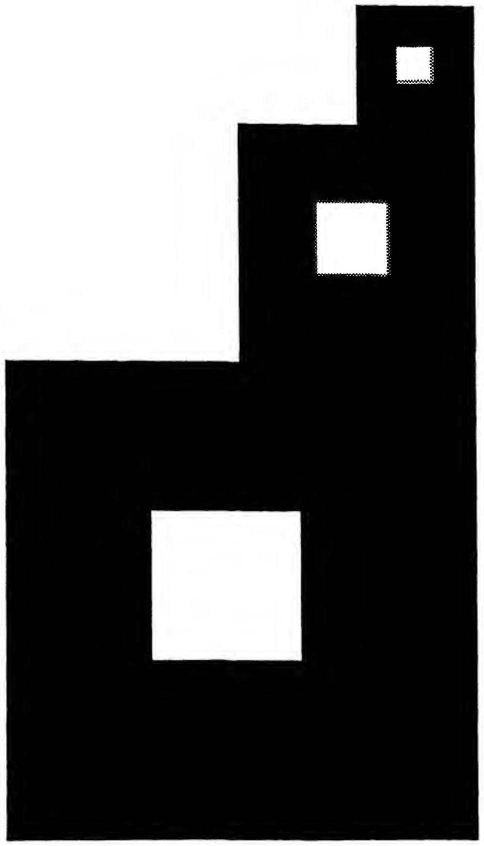


Figure 4.5: Gaussian Pyramid of Second Image Frame

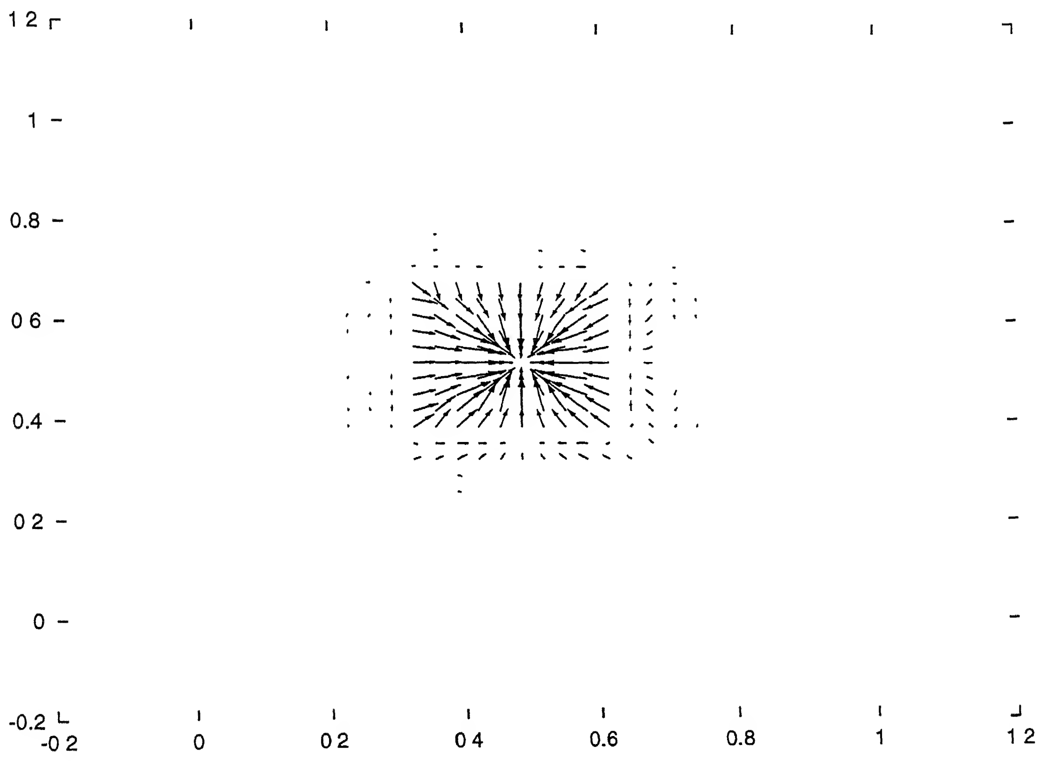


Figure 4.6: Optical Flow Vectors (at Level3) : The square is decreasing in size

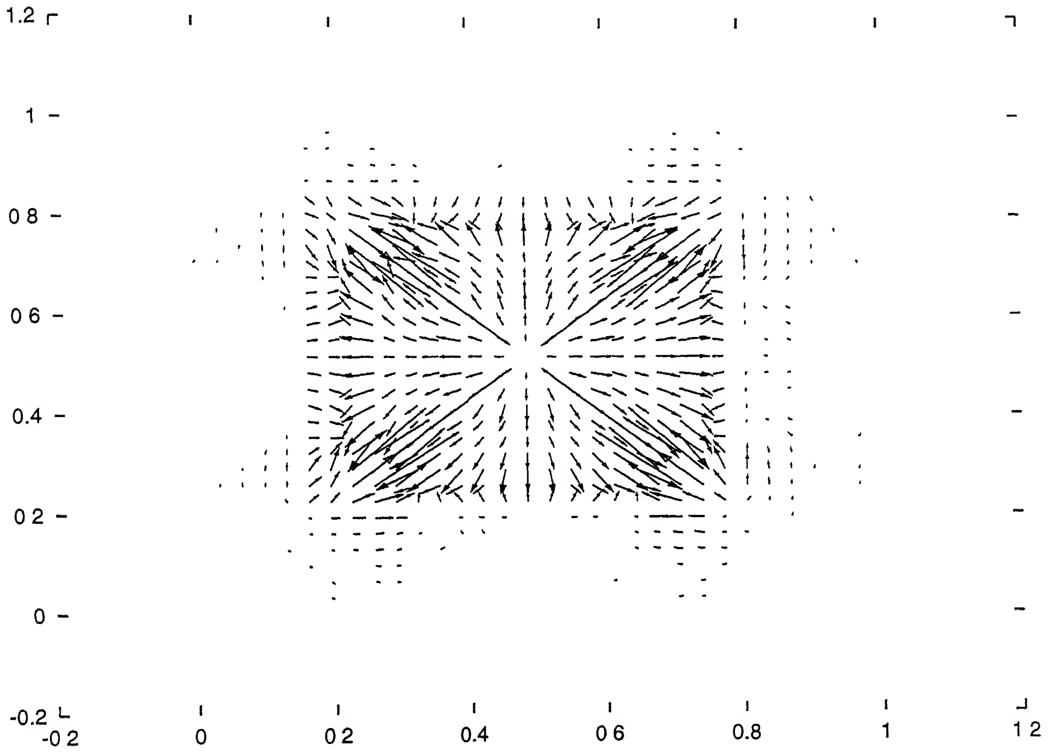


Figure 4.7: Optical Flow Vectors (at Level3) when the frames are interchanged: square is increasing in size

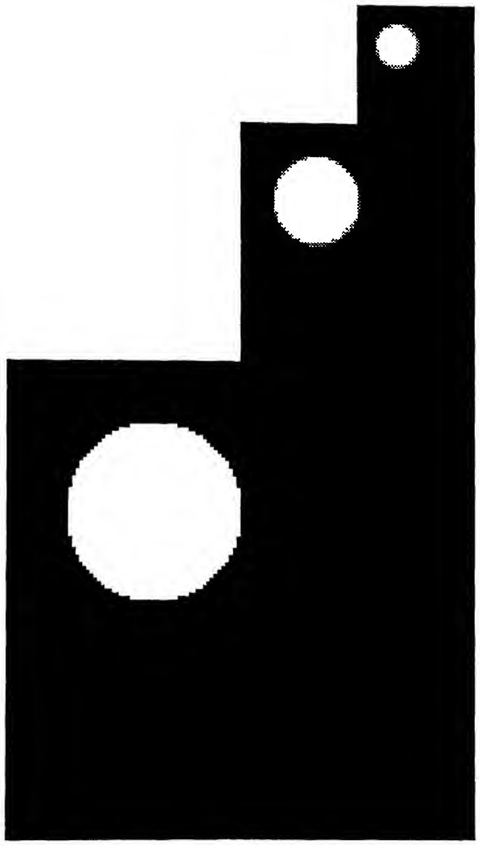


Figure 4 8: Gaussıan Pyramid of First Image Frame

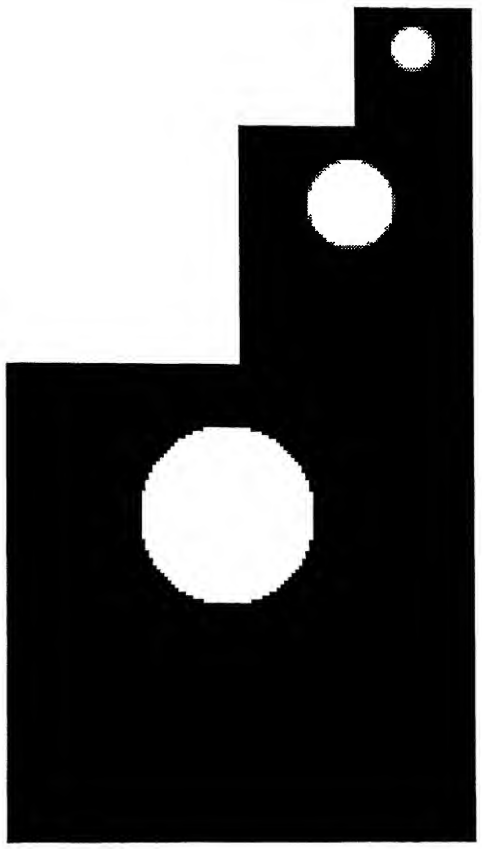


Figure 4.9: Gaussian Pyramid of Second Image Frame

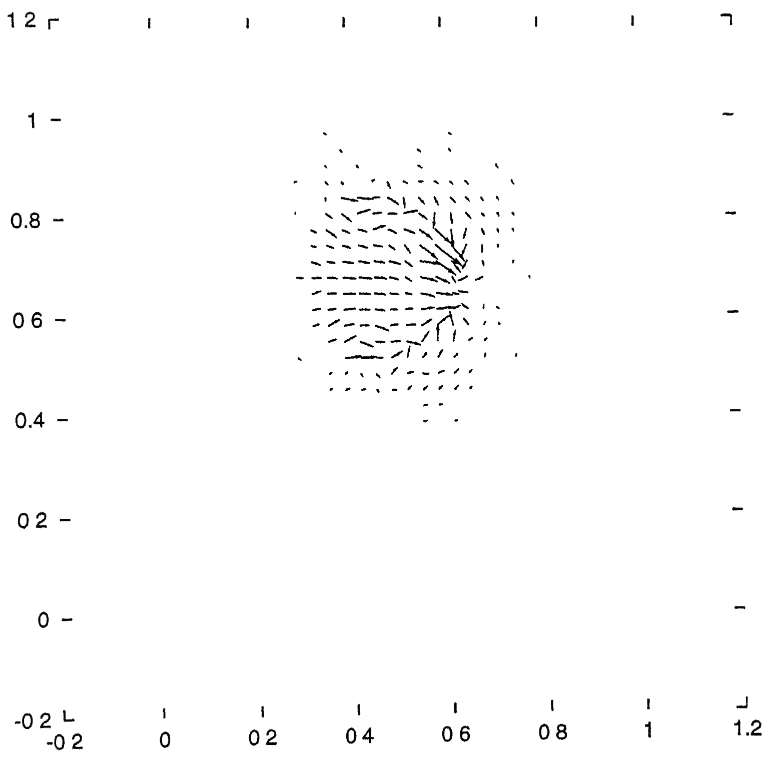


Figure 4.10: Optical Flow Vectors (at Level3)

Chapter 5

Conclusion

In this thesis we developed a TMS320C40 DSP based real time image processing system. The evaluated speed performance showed that it is capable of processing a large amount of data thus making it well suitable for implementing Hierarchical algorithms. We have considered implementation of hierarchical algorithms for line extraction, texture discrimination and optical flow estimation on the configured system. In the hierarchical algorithm for line extraction, problems which occur in the standard HT such as the formation of insignificant groups are avoided . Figure 4.2 shows the result of the Line Extraction algorithm applied to the image containing the cube. The extracted image shows that the straight edges are clearly delineated from the image. The groups at the corners are not propogated up the hierarchy and hence there is error at the corners of the cube. A texture discrimination algorithm based on the T.E.M. has been discussed. This algorithm has the advantage that it employs fixed size masks. It provides an initial decomposition of the image, which is used for further texture segmentation. The result in figure 4.3 shows that the central square block(one texture) is clearly discriminated from the background(another texture) in the T.E.M.. The optical flow estimation algorithm which used the interlevel motion smoothness constraints and the multigrid technique has been described in detail. The multigrid technique increased the convergence speed of the algorithm, making it well suitable for real-time image processing application. Figure 4.4 and 4.5 shows the Gaussian pyramid of the first frame and the second frame respectively of the image of a square block reducing in size between the two frames. The flow vectors in figure 4.6 clearly indicates the direction of the motion of the square block. The error along the boundary regions is due to the edge effect which is always found in all the gradient based methods. Figure 4.8 and 4.9 shows the Gaussian pyramid of the first frame and the second frame respectively of the image of a ball moving towards the right and figure 4.10 shows the Flow vectors. The Flow vectors of level 3 are plotted for the sake of clarity.

Future Scope of Work

The work on the line extraction can be extended for estimation of motion [18]. This is a new approach where the objects are uniquely represented by straight lines along the boundary and the motion is estimated from the shifts in ρ, θ parameters between the two consecutive motion frames. The Optical flow estimation can be used in robot vision(smart sensing) for object tracking. Here the flow vectors could be used as the information for the robot so that the robot can adjust its vision according to the target motion. Another extension to the work on flow estimation could be Video coding. Here the flow vectors are used for motion compensation to remove the temporal redundancies. Then the motion compensated error is subjected to D.C.T.(Discrete Cosine Transform) [19]. The error, after decoding, is transmitted.

Bibliography

[1] S. Mallat. Wavelets for a vision. *Proc. I.E.E.E.*, 84:604–614, 1996.

[2] Illingworth J Princen, J. and Kittler J. Hierarchical approach to line extraction based on hough transform. *Computer Vision, Graphics and Image Processing*, 52:57–77, 1990.

[3] J.M. Jolian and A. Rosenfeld, editors. *A pyramid framework for early vision.* Kluwer Acadamic, Boston, 1994.

[4] A. Rosenfeld, editor. *Multiresolution Image Processing and Analysis.* Springer-Verlag, New York, Tokyo, 1984.

[5] *The OCULUS-F/64 Frame Grabber User's Manual.* Coreco Inc, 1994.

[6] *SPIRIT-40 ISA User's Guide.* Sonitech International Inc, 1995.

[7] *TMS320C4x User's Guide.* Texas Instruments, 1994.

[8] Burt P.J. Smart sensing within a pyramid vision machine. *Proceedings of the I.E.E.E.*, 76(8):1006–1015, 1988.

[9] Luo Y. Chin F., Choi A. Optimal generating kernels for image pyramids by piecewise filtering. *I.E.E.E. Trans. Pattern Analysis Mach. Intell.*, 14:1190–1198, 1992.

[10] A.K. Jain, editor. *Fundamentals of Digital Image Processing.* Prentice-Hall of India, New Delhi, 1989.

[11] Salari E. and Ling Z. Texture segmentation using hierarchical wavelet decomposition. *Pattern Recognition*, 28(12):1819–1824, 1995.

[12] S.H. Hwang and S.U. Lee. A hierarchical optical flow estimation algorithm based on interlevel motion smoothness constraint. *Pattern Recognition*, 26:939–952, 1993.

[13] Horn B.K.P. and Schunck B.G. Determining optical flow. *Artif. Intell.*, 17:185–204, 1981.

[14] Aggarwal J.K. and Nandhakumar. On the computation of motion from sequences of images-a review. *Proceedings of the I.E.E.E.*, 76(8):917–935, 1988.

[15] D. Terzopoulos. Image analysis using multigrid relaxation methods. *I.E.E.E. Trans. Pattern Analysis Mach. Intell.*, 8:129–139, 1986.

[16] Glazer F. Multilevel relaxation in low-level computer vision. In Rosenfeld A., editor, *Multiresolution Image Processing and Analysis*, pages 312–330. Springer-Verlag Berlin Heidelberg New York Tokyo, 1984.

[17] Gerald C.F. and Wheatley P.O., editors. *Applied Numerical Analysis*. Addison-Wesley, Reading, Massachusetts, 1989.

[18] Li Hsiang-Ling and Chakrabarti C. Motion estimation of two-dimensional objects based on the straight line hough-transform:a new approach. *Pattern Recognition*, 29(8):1245–1258, 1996.

[19] Didier J. Le Gall. The mpeg video compression algorithm. *Signal Processing: Image Communication*, 4(2).129–140, 1992.

[20] Lee Seong-Whan. Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network. *I.E.E.E. Trans. Pattern Analysis Mach. Intell.*, 18(6):643–652, 1996.